FAME User Guide: Configuration and Installation

This user guide explains how to install and configure the Apache2-Perl module FAME and use it for authenticating users on the IdP (Identity Provider) side in the Shibboleth infrastructure in order to help control access to the resources published by SPs (Service Providers). The guide applies to configuring a Shibboleth IdP together with FAME module to be used for authenticating users and providing the additional LoA (Level of Assurance) attribute to the requesting SPs.

The FAME module is configured via directives that should be placed in the Apache configuration file (usually httpd.conf) and, thus, this document should be read in conjunction with the standard Apache Web Server documentation (available at http://httpd.apache.org/docs/2.0/). The FAME module also uses an external relational database to keep information about its Authentication Servers, users and secret keys.

It is presumed that you already have a working installation of Shibboleth IdP, i.e. that you have installed the Apache Web Server, Tomcat servlet container, mod_jk Tomcat-Apache plug-in that handles the communication between Tomcat and Apache, the Shibboleth IdP servlet, and LDAP Directory for storing Shibboleth attributes.

0 FAME Prerequisites and Installation Kit

The configuration/installation instructions described in this document are given with reference to the Linux/Unix environment. Before you start to install and configure FAME, the following gives a detailed list of the required prerequisite components:

(1) Apache Web Server version 2.0, available from http://apache.org/;
(2) Perl interpreter (version 5.8.6 was used for the development of FAME), available from http://www.perl.com/;
(3) mod_perl – an Apache module for providing a persistent Perl interpreter embedded in the Web Server for creating Apache-Perl modules, available from http://apache.perl.org/;
(4) The following Apache-Perl modules are required for the correct functioning of FAME: Apache2::Request, Apache2::RequestRec, Apache2::RequestIO, Apache2::RequestUtil, Apache2::ServerRec; Apache2::ServerUtil, Apache2::Connection, Apache2::Log, Apache2::Const, Apache2::Cookie, APR::Table, APR::Const, ModPerl::Registry, ModPerl::Util, Crypt::CBC, Crypt::Rijndael, Crypt::Random, Digest::MD5, MIME::Base64, IO::File, DBI, Net::LDAP, all available from http://www.cpan.org/.
(5) Mysql database version 4.1 or above, or a similar relational database system.

We have used Shibboleth IdP version 1.3 when developing the FAME system, available from http://shibboleth.internet2.edu/.

The FAME installation kit comprises the following items:

(1) The source code *Fame.pm* of the Apache-Perl FAME module;
(2) The folder called *fame* containing two sub-folders: *images* and *css* that respectively contain images and a style-sheet used by the FAME module for rendering HTML pages;

(3) The database set-up script *fame.sql* that can be used to create the FAME database;
(4) An example Apache-Perl script *AS.pm* for setting up an Authentication Server in conjunction with the Apache Web Server.
(5) An LDAP schema file *fame.schema* needed for the integration with Shibboleth's LDAP store.
(6) A report on the design of FAME (it is advised that you read this design document before starting to configure FAME).
(7) FAME User Guide, i.e. this document;

In order to configure a Shibboleth IdP in conjunction with FAME, the following things should be done:

(1) Configure Apache2 Server to use the FAME module;
(2) Create the FAME database;
(3) Set up the Authentication Server(s) and configure FAME to use them.
(4) Integrate FAME and Shibboleth

The following four sections address each of the above points.

1 Configuring FAME and Apache

The FAME module has been developed for and tested with Apache2, and it is assumed that an Apache2 Server with a support for Perl (via mod_perl) has been previously installed. The FAME system is implemented as an Apache-Perl module called *Fame.pm* (the source code of which can be found in the installation kit). The module is created under the *MyApache2*:: namespace, thus its full name is *MyApache2::Fame.pm*.

Locating the module

First you have to create a directory where the Fame module will reside in. For example, create directory called *perl* within your Web Server root (on our system */etc/apache2*), where you will keep all your Perl modules, i.e. create directory */etc/apache2/perl*. As the module is created under the *MyApache2*:: namespace, create the *MyApache2* subdirectory within the *perl* directory. The *MyApache2* subdirectory is where the *Fame.pm* module should be located.

Next, you need to tell Apache where to look for this module. Apache's *mod_perl* can be configured to invoke a start-up file (typically called *startup.pl*) with a set of Perl commands each time the server is launched or restarted. This is where we place the '*use lib*' statement that will instruct Apache where to find *Fame.pm*. You may also include configuration directives for the FAME module in *startup.pl* after the '*use lib*' command, or, alternatively, these directives can be configured via *httpd.conf*. If you choose to configure the module in *startup.pl*, the following gives an example of how this can be done (the commands related to the FAME module are at the bottom and highlighted in green). Place *startup.pl* in the directory containing all other configuration files for the Apache modules (i.e. files ending with *.conf*). Meaning of the FAME configuration directives used in the example will be explained shortly.

```
File: /etc/apache2/modules.d/startup.pl
```

```
use lib qw(/home/httpd/perl);

use ModPerl::Util (); #for CORE::GLOBAL::exit

use Apache2::RequestRec ();
use Apache2::RequestIO ();
use Apache2::RequestUtil ();

use Apache2::ServerRec ();
use Apache2::ServerUtil ();
use Apache2::Connection ();
use Apache2::Log ();

use APR::Table ();

use ModPerl::Registry ();

use Apache2::Const -compile => ':common';
use APR::Const -compile => ':common';

# FAME-related commands
# Location of the FAME module
use lib '/etc/apache2/perl';

# Load FAME module before configuring it
use MyApache2::Fame ();

# Configure FAME module
Apache2::ServerUtil->server->push_handlers(PerlChildInitHandler =>
\&MyApache2::Fame::configure((
        'FameLoginServer' => '/fame_login_server',
        'FameLogoutHandler' => '/fame_logout',
        'FameAuthTimeout' => 480,
        'FameSSOTimeout' => 1,
        'FameDB' =>
'dbi:mysql:database=fls;host=localhost;port=3306',
        'FameDBUser' => 'flsuser',
        'FameDBPassword' => 'flspassword',
        'FameSecretsTable' => 'secrets:secret_key:date_time',
        'FameASTable' =>
'authservers:url:auth_type:saml_auth_id:loa:secret_key',
         'FameUsersTable' =>
'fameusers:ldap_id:ldap_attribute:alternative_id',
        'FameShibLDAPServer' => 'localhost',
        'FameShibLDAPPort' => '389',
        'FameShibLDAPDN' =>
'cn=Manager,dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk',
        'FameShibLDAPPassword' => 'lm^s7a*56',
        'FameShibLDAPBaseDN' =>
'ou=shib-users,dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk'
        )));

1;
```

Finally, copy the *fame* directory from the installation kit, containing folders *images* and *css*, to your Apache Web Server's document root (on our system it is */var/www/localhost/htdocs*). It is important to copy these directories in your Web Server's document root as that they have to be "visible" from the Web, since they used by the FAME module for generating its HTML pages.

Loading the module

In order to configure *Fame.pm* module to work with the Apache Web Server, the module must first be loaded before using any of the directives explained below. If you have not loaded the module in *startup.pl* as previously explained, this can be (and typically is) done in Apache's *httpd.conf*.

```
File: /etc/apache2/httpd.conf

...
# Load FAME module before configuring it
use MyApache2::Fame ();
...
```

Configuration directives

In order to configure the FAME module, the following three blocks have to be created in Apache's configuration file *httpd.conf* and configured through various FAME directives. The first block is to specify the F-SSO (FAME's Single Sign On Handler) as the access control handler for the location (i.e. url) of the Shibboleth's HS (Handle Service). By doing this, the F-SSO is set up to protect the url of the Shibboleth's HS and the authentication type for this location is set to 'Fame'. The following gives an exemplar setting of this block:

```
File: /etc/apache2/httpd.conf

...
# Location of the Shibboleth's HS
<Location /shibboleth-idp/SSO>
     AuthType Fame
     AuthName "Fame Authentication Service"
     PerlAuthenHandler Apache2::Fame::sso_checker
     require valid-user
 </Location>
...
```

The above block specifies the *Apache2::Fame.pm* module's *sso_checker( )* routine to handle the user authentication for all requests for the Shibboleth HS's url (tied to location /shibboleth-idp/SSO).

The second block that has to be defined specifies the location and settings for the F-LS (FAME Login Server) that handles the user authentication and creation of SSO cookies. It should look similar to the following:

```
File: /etc/apache2/httpd.conf

...
<Location /fame_login_server>
     SetHandler perl-script
     PerlResponseHandler Apache2::Fame::login_server
</Location>
...
```

The third block defines the (optional) FAME logout handler. If, for any reason, the user wishes to terminate his current SSO session without closing his Web browser, he may do so by clicking the "logout" button on the main FAME page. This will redirect the user to the FAME logout handler, configured below.

```
File: /etc/apache2/httpd.conf

...
<Location /fame_logout>
     SetHandler perl-script
     PerlResponseHandler Apache2::Fame::logout
</Location>
...
```

The Fame module can be configured via a number of configuration parameters, as shown in the first two blocks above. They can either be set up in the Apache-Perl start-up script *startup.pl* (explained previously) using the module's *configure()* procedure, or in Apache's configuration file *httpd.conf* by defining each FAME parameter individually using mod_perl's PerlSetVar directive. FAME's configuration parameters can be defined anywhere in the main body of *httpd.conf*, but outside the above three blocks. The following lists all FAME's configuration parameters.

```
File: /etc/apache2/httpd.conf

...
# FAME's configuration parameters
 PerlSetVar FameLoginServer /fame_login_server
 PerlSetVar FameLogoutHandler /fame_logout
 PerlSetVar FameDB dbi:mysql:database=fls;host=localhost;port=3306
 PerlSetVar FameAuthTimeout 1
 PerlSetVar FameSSOTimeout 480
 PerlSetVar FameDBuser flsuser
 PerlSetVar FameDBPassword flspassword
 PerlSetVar FameSecretsTable secrets:secret_key:date_time
 PerlSetVar FameASTable authservers:url:auth_type:loa:secret_key
 PerlSetVar FameUsersTable fameusers:ldap_id:ldap_attribute:alternativ_id
 PerlSetVar FameShibLDAPServer localhost
 PerlSetVar FameShibLDAPPort 389
 PerlSetVar FameShibLDAPDN cn=Manager,dc=example,dc=com
 PerlSetVar FameShibLDAPPassword your_secret
 PerlSetVar FameShibLDAPBaseDN ou=shib-users,dc=example,dc=com
...
```

Whichever of the two configuration methods you select to use (i.e. configuration via *startup.pl* and *configure()* procedure or via *httpd.conf* and PerlSetVar directives), bare in mind that the following precedence rules apply:

- If a directive is specified in both *httpd.conf* and start-up script *startup.pl*, the value from the *httpd.conf* will be used and will override any other value.
- Otherwise, if a directive is not specified either in *httpd.conf* or in *startup.pl*, the default value will be used, provided the parameter has a default value. Otherwise, the module will report an error.

The following gives a detailed explanation of the FAME module’s valid configuration parameters.

FameLoginServer

This parameter specifies the relative url of the Fame Login Server (F-LS), and corresponds to the second <Location> block specified in the *httpd.conf* above. This directive tells the F-SSO where to redirect the user if he/she has not been authenticated yet (detected by the absence of the **sso** cookie).

Default value: `/fame_login_server`.

FameLogoutHandler

This parameter specifies the relative url of the Fame Logout Handler. It is used to enable the users to logout and clear all FAME-set cookies, i.e. to reset the current SSO session.

Default value: `/fame_logout`.

FameAuthTimeout

This parameter specifies the number of minutes that the user is left to successfully complete authentication. It is computed as a difference between the creation time of an auth-control cookie (by F-LS) and the receipt of the auth-reply token (by the F-LS from the Authentication Server). If this timeout is expired, the user will be forced to log in (authenticate) again.

Default value: 1 minute.

FameSSOTimeout

This parameter specifies the number of minutes before the **sso** cookie is considered expired, i.e. the duration of the user’s SSO session. After that time, the user will be forced to re-authenticate.

Default value: 480 minutes (8 hours).

FameDB

This parameter specifies the url string for the Perl DBI (DataBase Interface) to use when connecting to the FAME database. FAME has been developed using Mysql database, but a number of other databases can be used with Perl DBI (see http://dbi.perl.org for details) and configured to work with FAME.

Format:
<*perl_database_inteface*>:<*database_type*>:database=<*database_name*>;host =<*host_name*>;port=<*port_number*>.

Default value: `dbi:mysql:database=fls;host=localhost;port=3306`.

If the host is ‘localhost’and the port is ‘`3306`’, they can be omitted from the FameDB string.

FameDBUser

This parameter specifies the *username* to use when connecting to the FAME database above. The specified use must have read privileges for the FAME database.

Default value: `flsuser`.

FameDBPassword

This parameter specifies the password to use when connecting to the FAME database.

Default value: `flspassword`.

FameSecretsTable

This parameter specifies the name of a table containing secret keys used by the F-LS and F-SSO, as well as the names of the two columns of this table used for the secret key itself and the secret key version. Typically, a timestamp is used as a secret key version number when the key is inserted in the database.

Format:

<*table_name*>:<*secret_key_column_name*>:<*secret_key_version_column_name*>.

Default value: `secrets:secret_key:date_time`.

FameASTable

This parameter specifies the name for the table containing information about Authentication Servers, and the names of the five columns of this table that are used by FAME. Columns are: the url of the Authentication Server, the authentication type that the Authentication Server provides, the unique URN for the authentication method as defined by SAML1.1 (this is passed to the SP at the moment, but might be in future in order to pass the exact authentication method to the SP in addition to LoA), the LoA of the Authentication Server and the secret key shared between the AS and F-LS (in Base64 format).

Format:
<*table_name*>:<*url_column_name*>:<*auth_type_column_name*>:<*saml_auth_id*>:<*loa_column_name*>:<*secret_key_column_name*>.

Default value:

`authservers:url:auth_type:saml_auth_id:loa:secret_key`.

FameUsersTable

This parameter specifies the name of the table containing mapping between the FAME users' ids and their corresponding DNs in the Shibboleth LDAP directory. Each FAME user has a unique entry in the Shibboleth's LDAP directory, identified by his LDAP DN, where his current LoA value is stored and picked up by Shibboleth. However, each FAME user may have several identities, e.g. in cases when their IdP supports several methods of authentication, none of which needs to use LDAP to store users' credentials. For this reason, we have to map these different identities of the same user to the user's identity in the Shibboleth's LDAP directory. FameUsersTable contains such mappings, i.e. it consists of three columns: the user's LDAP id (part of the user's LDAP DN), the name of the LDAP attribute which keeps the user's id (e.g. *uid* or *cn*), and the user's alternative id (e.g. Kerberos principal name such as *kerbuser@CS.MAN.AC.UK*, or subject of the public key certificate such as *C=GB/ST=Lancashire/L=Manchester/O=University of Manchester/OU=School of Computer Science/CN=Alex Nenadic/emailAddress=anenadic@cs.man.ac.uk*, or any other user's alternative username).

Format: <*table_name*>:<*ldap_id_column_name*>:<*ldap_attribute_column_name*>:<*alternative_id*>.

Default value:

`fameusers:ldap_id:ldap_attribute:alternative_id`.

Note that, even if the IdP utilises LDAP for user authentication, all LDAP users must be inserted in the FameUsersTable. In this case, the <*ldap_id*> and <*alternative_id*> fields would contain the same values. As mentioned before, a single user may have multiple entries in this table, each corresponding to his identities with different Authentication Servers. However, all these entries must be linked to the user's identity in the Shibboleth's LDAP directory.

FameShibLDAPServer

This parameter specifies the name (or IP address) of the LDAP Server Shibboleth uses for storing user attributes.

Default value: `localhost`.

FameShibLDAPPort

This parameter specifies the port the Shibboleth LDAP Server is running on.

Default value: `389`.

FameShibLDAPDN

This mandatory parameter specifies the distinguished name of the user used to bind to the Shibboleth LDAP directory. The user must have the write privileges for the 'loa' attribute stored in the directory.

Example: `'cn=Manager,dc=example,dc=com'`.

Default value: `none`. The Fame module will attempt an anonymous bind if this item is not configured. It will almost certainly fail when an update to the 'loa' attribute is attempted later on, so it is strongly advised to set up this parameter.

FameShibLDAPPassword

This mandatory parameter specifies the password for the above DN.

Default value: `none`.

FameShibLDAPBaseDN

This mandatory parameter specifies the sub-tree of the Shibboleth LDAP directory where searches for the users should start from.

Example: `'ou=shib-users,dc=example,dc=com'`.

Default value: `none`.

2 FAME Database Configuration

The FAME database consists of three tables: the Secrets table, the Authentication Servers table, and the FAME users table. A database user with read-only access to this database is required for the use by the Fame module. An exemplar database called *fls* and a database user *flsuser* (with password *flspassword*) can be created as follows.

```
# Create FAME database
CREATE DATABASE fls;
GRANT SELECT on fls.*
TO flsuser IDENTIFIED BY ‘flspassword’;
```

The names used for the database, the Secrets, Authentication Servers and FAME users tables, database user and password can be respectively set up for the FAME module using configuration parameter *FameDB, FameSecretsTable, FameASTable, FameUsersTable, FameDBUser* and *FameDBPassword.*

Secrets table

The Secrets table stores the secret keys used by the F-LS and the F-SSO for creation of cookies passed between them (namely, the **sso** and the **request-url** cookies). Each key has a version number associated with it, which is implemented as the timestamp when the key was inserted into the database. The key versioning allows periodical updates of the secret key without invalidating current valid cookies created with previous keys. This table is configured via parameter *FameSecretsTable* and can be created as follows.

```
# Create secrets table
CREATE TABLE secrets (
  secret_key text NOT NULL,
  date_time datetime NOT NULL default '0000-00-00 00:00:00'
);
```

Authentication Servers table

The Authentication Servers table stores information about configured/supported Authentication Server(s). Each row includes the url at which the AS is running, the authentication type (exemplar settings are Username/password, Kerberos, Browser certificate, Smart-card certificate) and are shown on the main FAME page as an option for the user to choose, the LoA provided by the AS, the secret key shared between the AS and the F-LS (which is used for encryption and decryption of the **auth-request** and **auth-reply** tokens), and an optional timestamp. The last (i.e. timestamp) column in the table is optional, as it is not used by the FAME module. This table is configured via parameter *FameASTable* and can be created as follows.

```
# Create Authentication Servers table:

CREATE TABLE authservers (
  url varchar(100) NOT NULL default '',
  auth_type varchar(100) NOT NULL default '',
  saml_auth_id varchar(100) NOT NULL default NULL,
  loa smallint(6) NOT NULL default '1',
  secret_key text NOT NULL,
  date_time datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (url)
);
```

FAME Users table

The Users table stores information about mappings between various identities the FAME users hold with different Authentication Servers and their identity in the Shibboleth LDAP directory, where the user's current LoA is stored and later picked up by Shibboleth. Each row contains the user's LDAP id, the name of the LDAP attribute that this id refers to (such as *uid*, or *cn*) and the user's alternative id with an Authentication Server set up by the IdP. This table is configured via parameter *FameUsersTable* and can be created as follows.

```
# Example:

CREATE TABLE fameusers (
  ldap_id varchar(255) NOT NULL default '',
  ldap_attribute varchar(100) NOT NULL default '',
  alternative_id varchar(255) NOT NULL default ''
);
```

The *fame.sql* script found in the installation kit can be used to automatically create the FAME database called *fls* and the default database user *flsuser* (password *flspassword*). If you want to use a different username/password for the database user or different names for the database, table or column names (other than those used in the script), modify the script to reflect this and make sure you pass the correct values to the FAME module via corresponding configuration parameters. To execute the *fame.sql* script, go to the directory where your *fame.sql* script is located and type in the following.

```
$ mysql -u root –p < fame.sql
Enter password:
```

To verify that *fame* database has been created properly, connect to mysql database:

```
$ mysql -u root –p
Enter password:
```

Type in the administrator's password and you should receive the mysql prompt.

```
mysql>
```

Type in the following command to see the list of databases:

```
mysql> show databases;
```

You should be seeing something like the following:

```
mysql> show databases;
+------------+
| Database   |
+------------+
| fls        |
| mysql      |
| test       |
+------------+
5 rows in set (0.00 sec)

mysql> use fls;

Database changed
mysql> show tables;
+---------------+
| Tables_in_fls |
+---------------+
| authservers   |
| fameusers     |
| secrets       |
+---------------+
3 rows in set (0.00 sec)
```

3 Authentication Server(s) Configuration

The FAME system attempts to use existing authentication systems and to integrate with them with minimum modifications. We distinguish two cases of integration based on how the Authentication Server (AS) is implemented.

(1) If AS is a standard Apache module for protecting Web resources (such as Kerberos system using Apache module *mod_auth_kerb* or an LDAP-based authentication system using *mod_auth_ldap*), then it is only required to install a script (*AS.pm*) provided in the FAME installation kit in order to make the FAME interoperate with the AS and no alterations to the AS are required.

(2) If AS is a custom-built system, then some modifications are required on the AS's side, in order to accommodate requests passed by the F-LS component of the FAME system and return the necessary information back to the F-LS upon authentication. In this case, it is necessary to understand how information is passed between the F-LS and the AS in order to make the necessary modifications to the AS.

The AS receives the authentication request from the F-LS in the form of:

https://<address_of_the_AS>?AuthRequestToken=<encrypted_auth_request_token>

This means that the encrypted **auth-request** token is passed to the AS as an url parameter. It contains two parameters and is encrypted by the F-LS with the FLS_AS_KEY, a symmetric key shared between the F-LS and AS, before it is passed to the AS. The AS needs to decrypt the token and extract the two parameters contained in it: the random challenge (*RC*) and the return address of the F-LS (i.e. to where to redirect the user upon successful authentication), which are delimited by a comma (","). The format of the **auth-request** token is:

<auth_request_token> = <random_challenge>,<address_of_the_FLS>

<encrypted_auth_request_token> = $E_{FLS\_AS\_KEY}$(*< auth_request_token>*)

Upon successful authentication, the AS is required to redirect the user back to the F-LS and pass the **auth-reply** token via URL, which looks like the following:

https://<address_of_the_FLS>/?AuthReplyToken=<encrypted_auth_reply_token>

Auth-reply token is encrypted by the AS with the same shared symmetric key FLS_AS_KEY. It contains the AS's response to the random challenge from the **auth-request** token (i.e. *RC* + 1) and the name of the authenticated user that has to be passed to Shibboleth (refer to Section 4.1 for information on what is considered as the user name), delimited by a colon (":"). The **auth-reply** token has the following format:

<auth_reply_token> = <random_challenge_response>:<user_name>

<encrypted_auth_reply_token> = $E_{FLS\_AS\_KEY}$(*< auth_reply_token>*)

In the case you are using a standard Apache authentication module (such as *mod_auth_kerb* for Kerberos authentication), you should use the *AS.pm* script provided in the installation kit to get your AS interoperate with the F-LS component of FAME. As *AS.pm* script is within the same namespace as *Fame.pm* module (i.e. *Apache2::* namespace), copy *AS.pm* to the same directory where *Fame.pm* is located (in this installation guide it is */etc/apache2/perl/MyApache2/*).

Let us suppose you are running Kerberos Authentication Server using Apache module *mod_auth_kerb*. You should create a <Location> in your *httpd.conf* that is tied to our *AS.pm* script and protected by Kerberos. Such a section in your *httpd.conf* may look something like the following:

File: /etc/apache2/httpd.conf

```
...
# Location protected by Kerberos
<Location /kerb-auth>
        SSLOptions +StrictRequire
        SSLRequireSSL
        PerlResponseHandler MyApache2::AS
         AuthType KerberosV5
         AuthName "Kerberos Login"
        KrbAuthRealms CS.MAN.AC.UK
        Krb5Keytab /etc/apache2/apache2_kerb.keytab
        KrbMethodK5Passwd on
        KrbServiceName HTTP
        KrbVerifyKDC on
        require valid-user
</Location>
...
```

The above code defines a <Location> within your Apache Web server served by the script *AS.pm* and accessible from a Web browser as *https://<your_server>/kerb-auth* only by a user successfully authenticated by Kerberos and using an SSL-protected connection. The <Location> */kerb-auth* plays the actual role of your AS. The *AS.pm* script serving this <Location> will receive requests from the F-LS (but only if the user has been authenticated by the set method previously), perform the necessary tasks, and redirect the user back to the F-LS.

The *AS.pm* script can be reused for any authentication system other than Kerberos. The only thing that needs to be configured inside the script is the path to the file containing the Base64-encoded secret key shared between the F-LS and AS (look for variable $KEY_FH in the source code of *AS.pm* script).

In the case you have a custom build Authentication Server (i.e. the one not using Apache's standard authentication modules), you will have to modify it to enable it to receive requests forwarded by the F-LS and generate responses to the F-LS upon the user's successful authentication. If the case the user has not authenticated correctly, it is up to your Authentication Server to display the error message to the user and ask for re-authentication.

4 FAME and Shibboleth Integration

After you have installed and configured the FAME module according to the previous three sections, the integration of the FAME module with the Shibboleth's IdP should proceed according to the following steps:

(1) Extend the Shibboleth LDAP directory's schema to include definitions of the two FAME-defined attributes: the 'loa' attribute and its expiration time;
(2) Insert the <SimpleAttributeDefinition> element in Shibboleth's *resolver.xml* to define the 'loa' attribute and the corresponding <JNDIDirectoryDataConnector> element to tell Shibboleth how and from which source to pull the loa attribute.
(3) Modify the Shibboleth IdP's ARP (Attribute Release Policy) located in file *site.arp.xml* to specify which requesting SPs should the *loa* attribute be released to.

Extending the Shibboleths LDAP Directory Schema
The *inetOrgPerson* LDAP object class is widely used in LDAP directories to represents people within organisations and has been endorsed by Shibboleth to store users in its LDAP store. The FAME system has extended this object class by defining a new LDAP object class called *famePerson* that inherits all attributes from the *inetOrgPerson* class and additionally defines two new attributes: *nist-loa* and *nist-loaExpires*. The first attribute is used to store the current 'loa' value for the user while the second stores the expiration time for the current 'loa' value (which is equal to the duration of the authenticated user's SSO session). The prefix '*nist*' in the attribute

name is used to denote the NIST E-Authentication Guideline[1] that out loa attribute definition conforms to. Shibboleth IdP is subsequently configured to pick up the *nist-loa* attribute from the LDAP store and pass it over to the requesting SP. The *nist-loaExpires* attribute is currently not passed to the SP, but is included in the LDAP store as well for possible future use.

The *famePerson* object class and the *nist-loa* and *nist-loaExpires* attributes are defined in the LDAP schema file called *fame.schema*, which can be found in the FAME installation kit.

File: fame.schema

```
# FAME LDAP schema.
# The attribute types and object class in this schema include the
# specifications of the 'loa' and 'loaExpires' attributes and the
# specification of the 'famePerson' object class. The 'famePerson'
# object class is an extension of the general purpose 'inetOrgPerson'
# object class, and additionally contains the two newly defined
# attributes 'loa' and 'loaExpires'.

# LoA attribute definition.
attributeType ( 1.2.826.0.1.3344810.1.1.104 NAME 'nist-loa'
        DESC 'The Level of Authentication Assurance conforming to the
NIST E-Authentication Guideline'
        EQUALITY integerMatch
         ORDERING integerOrderingMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
         SINGLE-VALUE)

# LoA's expiration date attribute definition.
attributeType ( 1.2.826.0.1.3344810.1.1.106 NAME 'nist-loaExpires'
        DESC 'Expiration date for the current LoA attribute'
          EQUALITY caseExactMatch
          ORDERING caseExactOrderingMatch
          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
          SINGLE-VALUE)

# famePerson object class definifion.
objectClass ( 1.2.826.0.1.3344810.1.0.24 NAME 'famePerson'
        DESC 'Person that uses FAME for authentication'
        SUP inetOrgPerson
        MAY ( nist-loa $ nist-loaExpires )
        )
```

To add this new attribute and object definitions to the IdP’s LDAP directory, *fame.schema* has to be copied the `schema` directory of the LDAP installation (on our system `/etc/openldap/schema`), where other LDAP schemas are stored as well. Then the copied schema has to be included in the LDAP’s configuration by inserting the line #4 below in the IdP LDAP Server’s configuration file (typically `/etc/openldap/slapd.conf`):

File: /etc/openldap/slapd.conf

```
...
```

[1] NIST Special Publication 800-63, E-Authentication Guideline, available at http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf

```
include         /etc/openldap/core.schema
include         /etc/openldap/cosine.schema
include         /etc/openldap/inetorgperson.schema
include         /etc/openldap/fame.schema
...
```

Configuring Attribute Definitions and JNDI Data Connectors

The Shibboleth IdP acquires all the attributes it sends to an SP by using a specialised *attribute resolver* defined in the file *resolver.xml*. In order for an attribute to be sent to the SP, it has to be converted to a SAML-based XML format and included in the *resolver.xml* file in the form of the <SimpleAttributeDefinition> element. Next, a <JNDIDirectoryDataConnector> element has to be defined and referred to by the just created loa's <SimpleAttributeDefinition> element, in order to tell Shibboleth how to pull the 'loa' attribute from a data store (in our case an LDAP directory).

To define the <SimpleAttributeDefinition> element for the 'loa' attribute, insert the following in *resolver.xml*:

```
File: /usr/local/shibboleth-idp/etc/resolver.xml
...
<SimpleAttributeDefinition id="urn:oid:1.2.826.0.1.3344810.1.1.104"
sourceName="nist-loa">
                <DataConnectorDependency requires="directory"/>
</SimpleAttributeDefinition>
...
```

The above code defines the attribute whose unique URN-like name is "urn:oid:1.2.826.0.1.3344810.1.1.104" (derived from the nist-loa attribute's unique oid) and which uses a DataConnector with an id `"directory"` to obtain the attribute value. To define such a DataConnector, insert the following in *resolver.xml*:

```
File: /usr/local/shibboleth-idp/etc/resolver.xml
...
<JNDIDirectoryDataConnector id="directory">
        <Search filter="uid=%PRINCIPAL%">
               <Controls searchScope="SUBTREE_SCOPE"
returningObjects="false" />
        </Search>
        <Property name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory" />
        <Property name="java.naming.provider.url" value=
"ldap://rpc56.cs.man.ac.uk/dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk" />
        </JNDIDirectoryDataConnector>
...
```

The above DataConnector element has an id=`"directory"`, connects to an LDAP directory defined by the url `"://rpc56.cs.man.ac.uk/"` and the LDAP directory root `"dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk"`, and uses a search filter `"uid=%PRINCIPAL%"` to search for the users when searching for the `%PRINCIPAL%`'s (i.e. user's) attributes. Modify the <Property> element to correspond to your LDAP server's settings.

Configure Shibboleth IdP's Attribute Release Policy

The Shibboleth's ARP (Attribute Release Policy) determines which of the defined attributes finally gets released to which requesting SPs. It acts as a filter for the attributes stored in the LDAP directory – ARPs can only be used to release the attributes that are already stored in the LDAP directory and defined in *resolver.xml*; it can only be used to limit what information gets released to whom. On the other hand, the attribute must be defined in both *resolver.xml* and specified in the site's ARP in order for it to be passed to a requesting SP via Shibboleth.

The simplest configuration for the *loa* attribute is to define a site policy in *arp.site.xml* file. Policies stored in this file apply for the whole IdP's site, i.e. for every user for whom this IdP retrieves/releases information. In order to configure a simple policy to release the *loa* attribute to every requesting SP, the *arp.site.xml* should look like the following:

File: /usr/local/shibboleth-idp/etc/arps/arp.site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<AttributeReleasePolicy
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mace:shibboleth:arp:1.0"
xsi:schemaLocation="urn:mace:shibboleth:arp:1.0 shibboleth-arp-
1.0.xsd">
        <Description>Simplest possible ARP.</Description>
        <Rule>
                <Target>
                        <AnyTarget/>
                </Target>
                <!-- Loa Attribute -->
                <Attribute name="urn:oid:1.2.826.0.1.3344810.1.1.104>
                        <AnyValue release="permit"/>
                </Attribute>
        </Rule>
</AttributeReleasePolicy>
```